CpE 645 Image Processing and Computer Vision

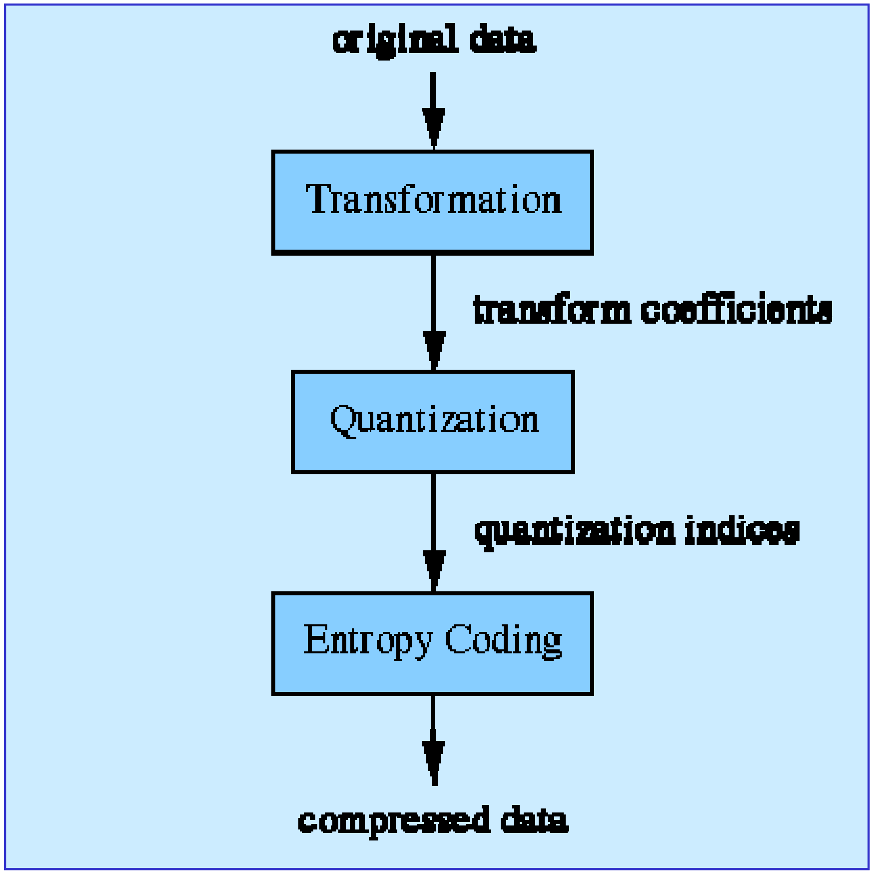
Prof. Hong Man

Department of Electrical and Computer Engineering
Stevens Institute of Technology

STEVENS Institute of Technology

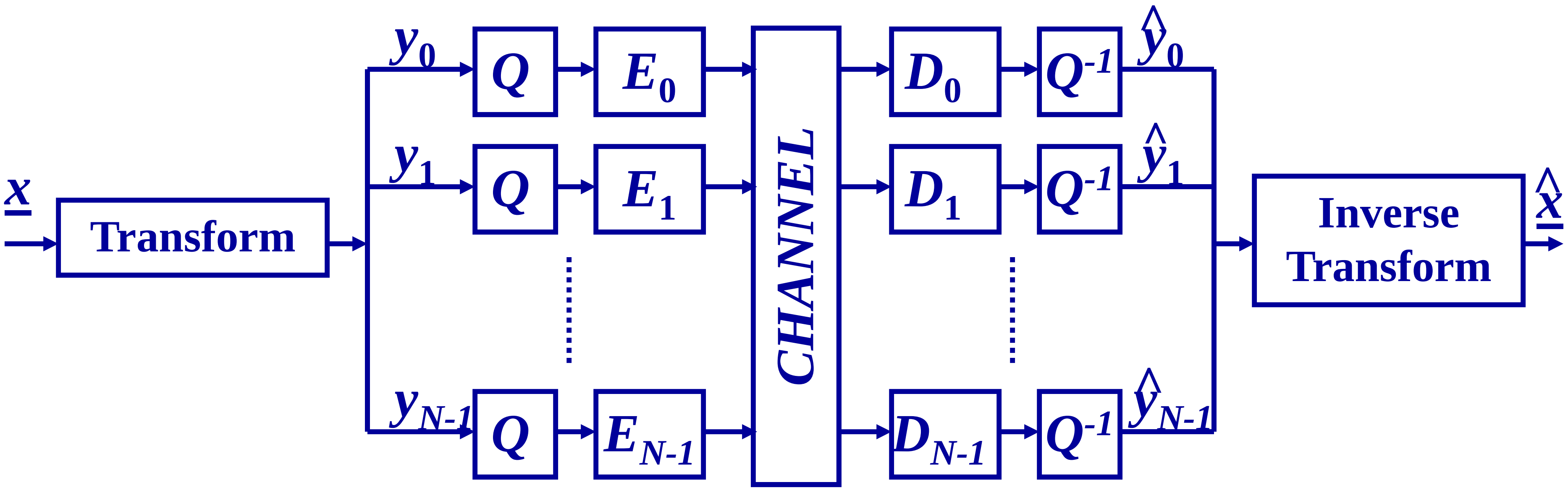
Visual Information Environment Laboratory

Elements of Image/Video Compression



Three stage image/video coding structure.

Transform Coding



Data Transformation

- Data transformation:

  represent an input data array by a new data array through an invertible data transform method.
- Why:
  - energy compaction,
  - de-correlation,
  - helpful data structure.
- How:
  - discrete cosine transform (DCT),
  - subband/wavelet transform.

Wavelet Transform

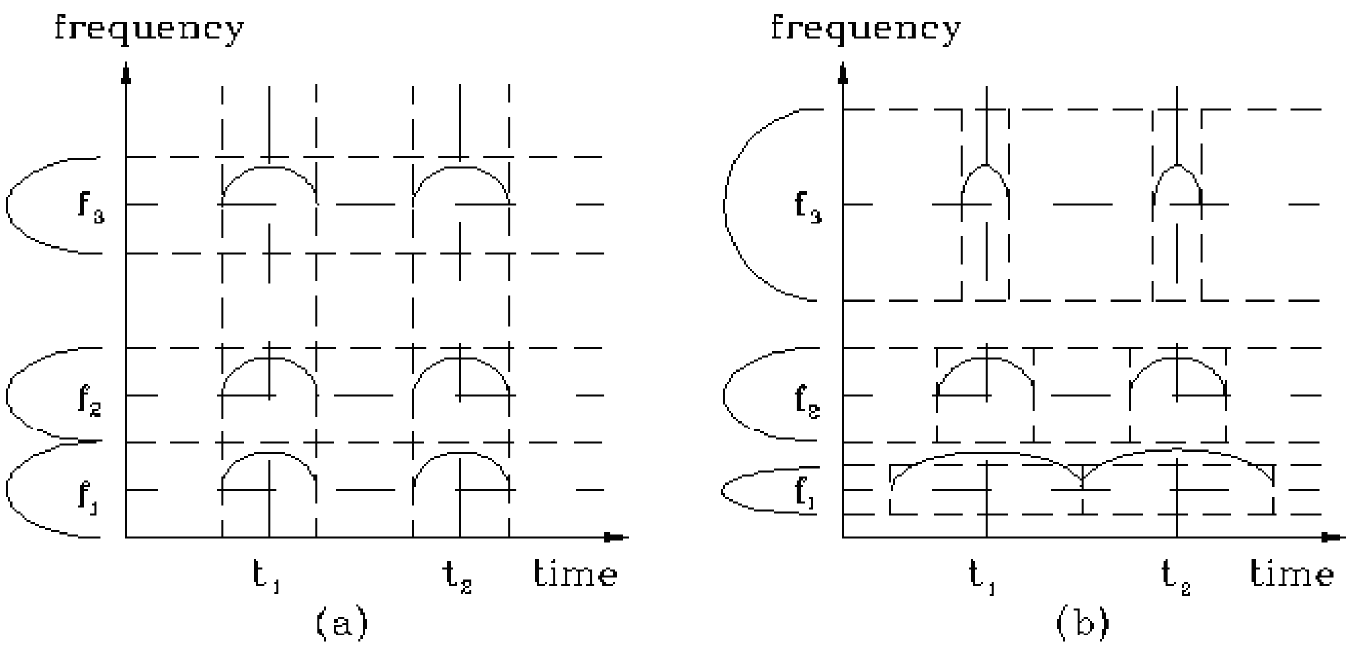
- Classical Fourier decompositions have good frequency resolution but poor time localization.
- Wavelet representations decompose the signal in terms of functions that are localized in both time and frequency.
- The basis functions of a wavelet transform are *wavelets*, which are dilated (scaled) and translated versions of a *mother wavelet*.
- Wavelets usually has finite space or time duration, which can provide localized information of a signal.
- Wavelets are also band limited in frequency, which can represent frequency components in a signal.

Wavelet Tutorial

- A nice wavelet tutorial

http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html

Wavelet Transform



(a) Short-time Fourier Transform (STFT)

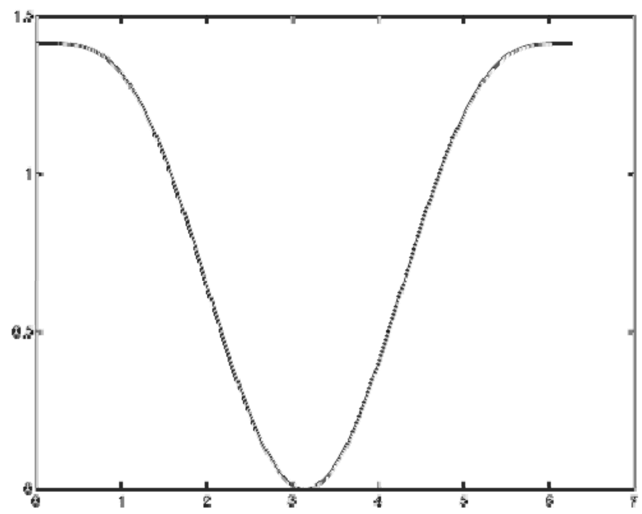
(b) Continuous wavelet transform

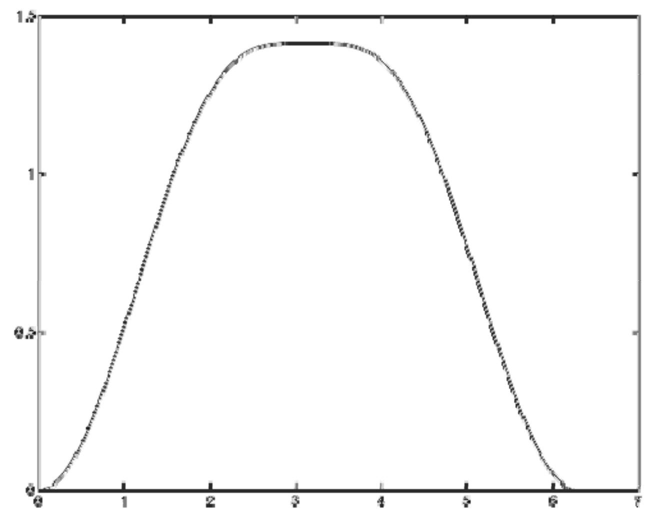
Wavelets

- A discrete wavelet transform (DWT) can be implemented through a 2-band subband filtering:

h_0=[0.48296291314453410,
0.8365163037378079,
0.2241438680420134,
-0.1294095225512604];

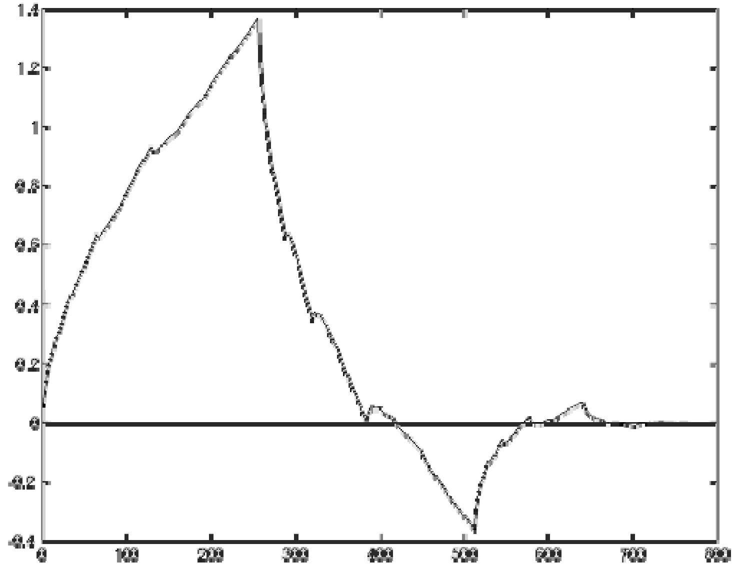
h_1=[0.1294095225512604,
0.2241438680420134,
-0.8365163037378079,
0.4829629131445341];

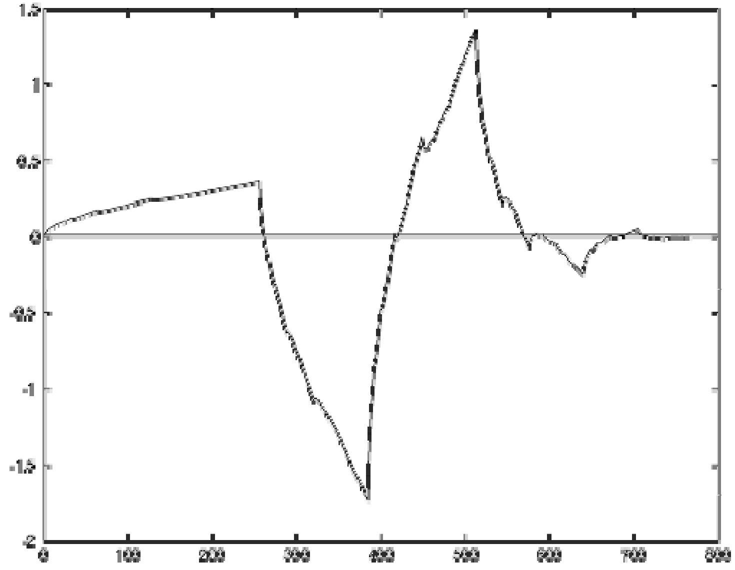




Wavelets

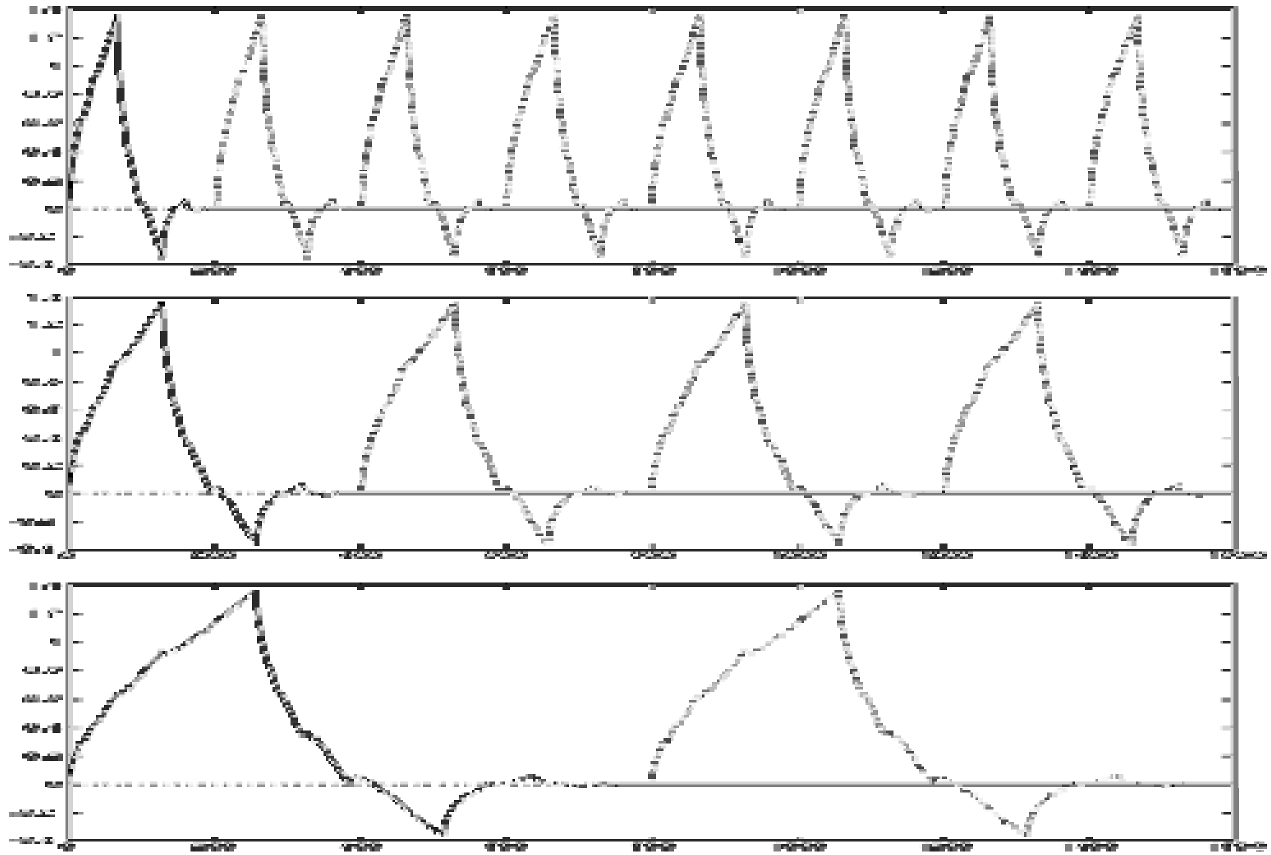
- The corresponding scaling function and wavelet function:





Wavelets

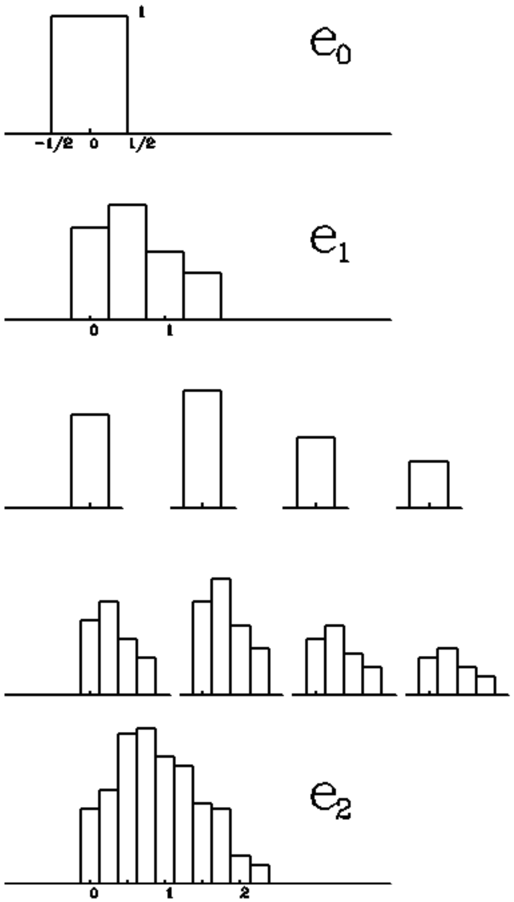
- Some dilated DWT basis functions:



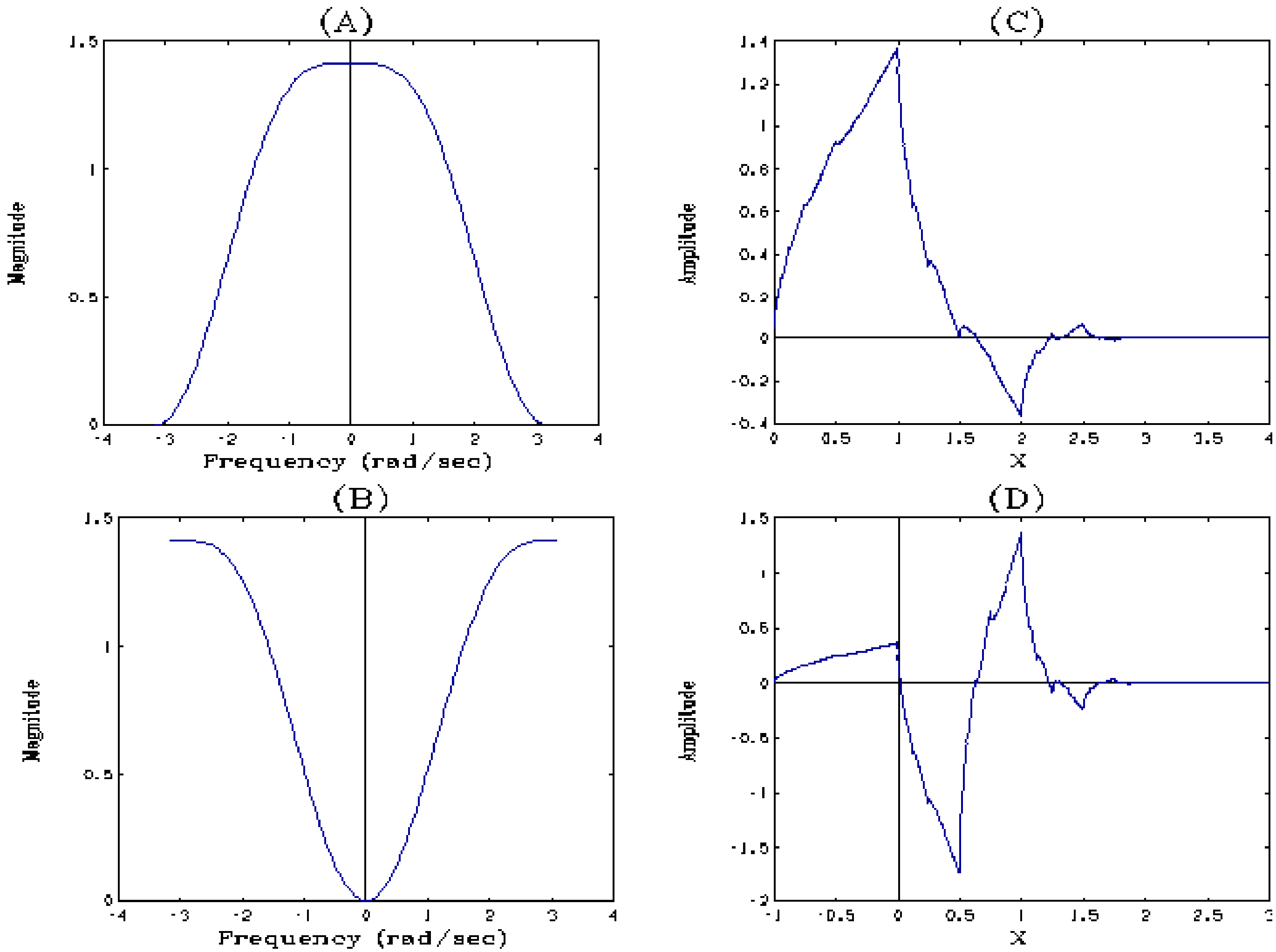
Visual Information Environment Laboratory

Wavelets

- A graphical recursion algorithm to associate filter banks with wavelets:
  1. Initialize a continuous unit square waveform.
  2. Pass it through the digital filters
  3. Represent each output sample as a new square waveform with half the time duration, and pass each of them through the filters,
  4. Linearly combine them to form the out put of this recursion, go back to step 3.



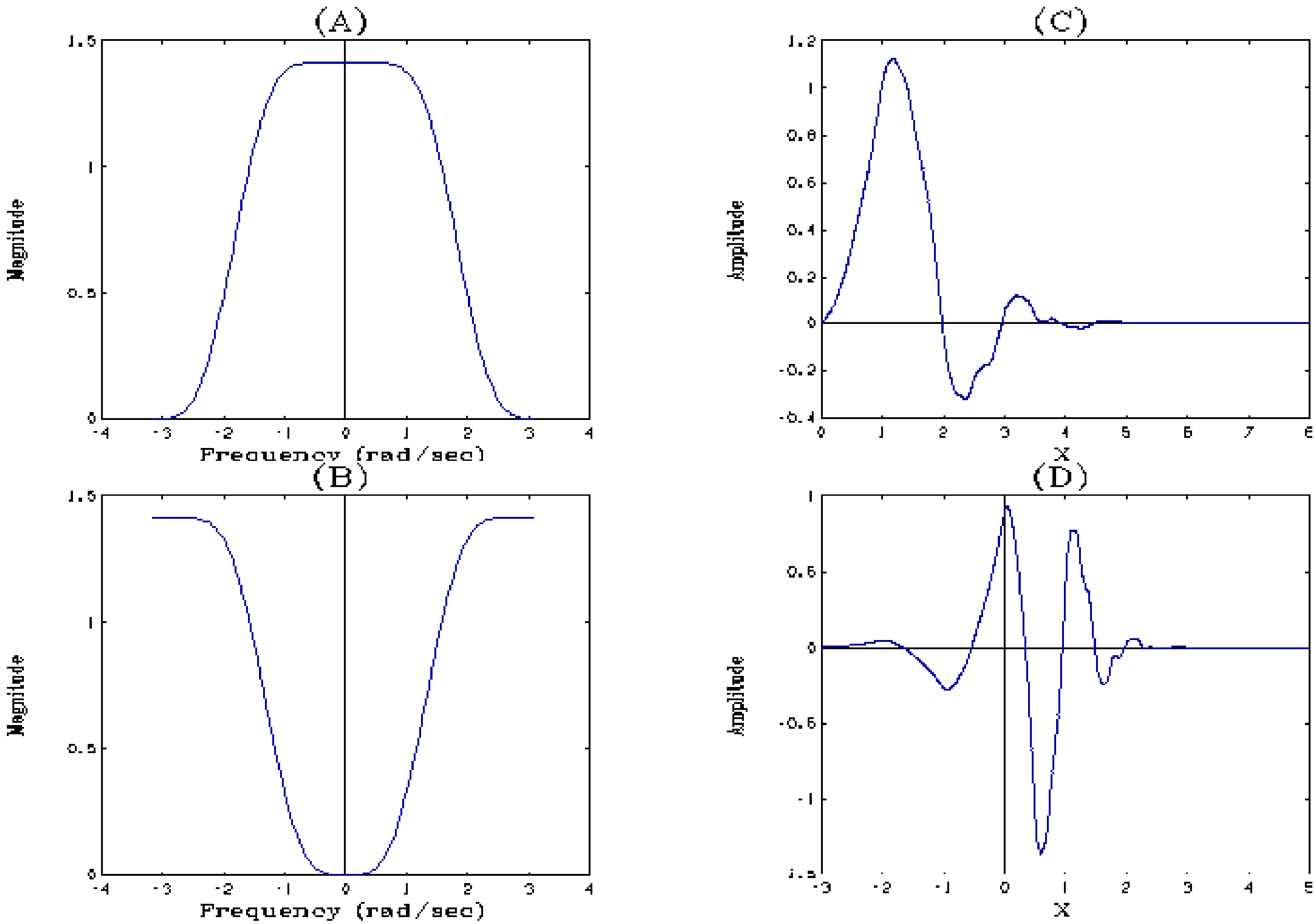
Wavelets with 4-tap FIR Filters



STEVENS Institute of Technology

Visual Information Environment Laboratory

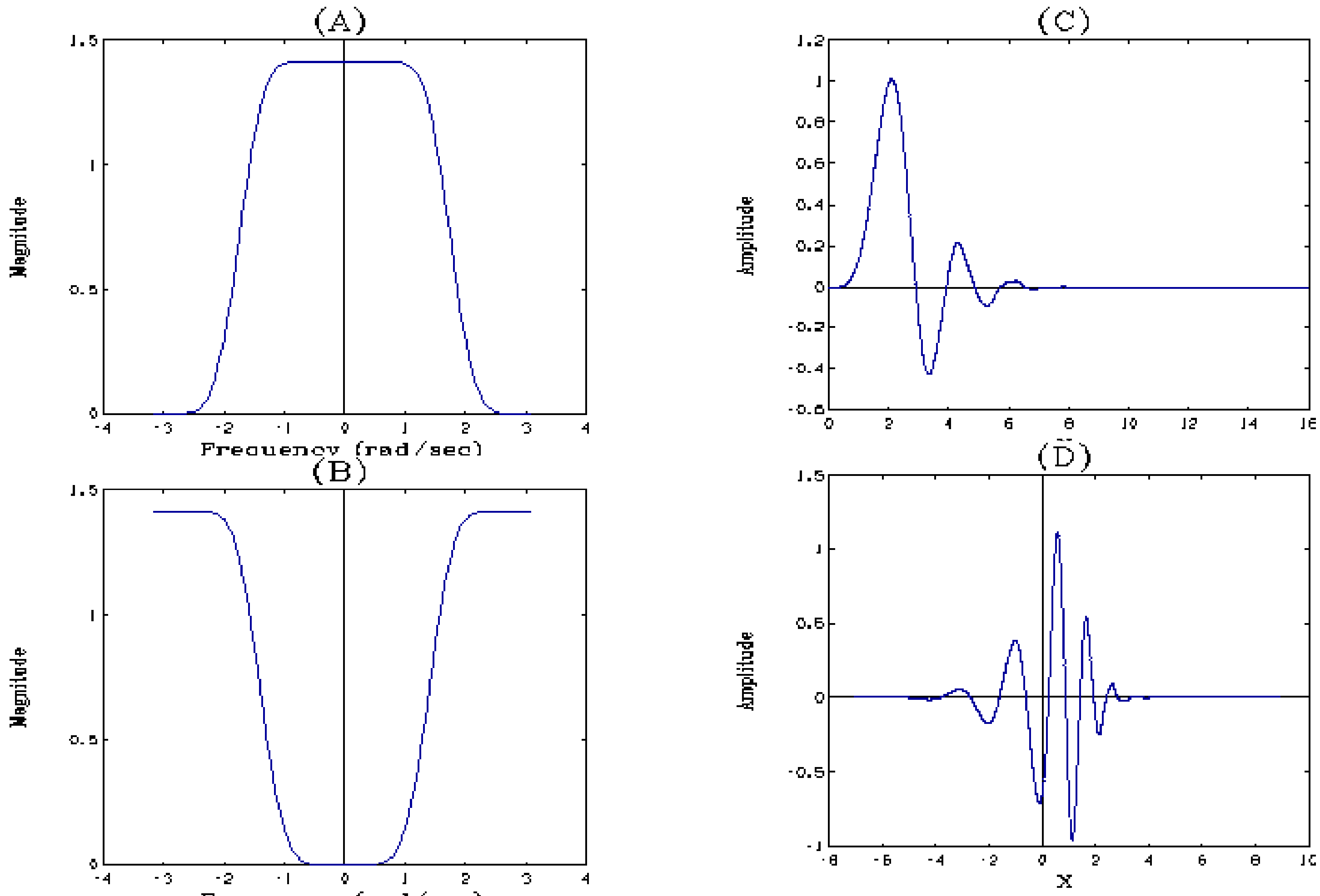
Wavelets with 8-tap FIR Filters



STEVENS
Institute of Technology

Visual Information Environment Laboratory

Wavelets with 16-tap FIR Filters



STEVENS Institute of Technology

Visual Information Environment Laboratory

Subband Coding

- Subband Coding is to decompose a signal into components by applying frequency-selective filtering. Then select the best coding technique that best suits each component (subjectively and objectively).
- *Example*: decompose a signal into slow- and fast-varying components:

  $$y[n-1]=x[n]+x[n-1], \quad y[n]=x[n]-x[n-1].$$

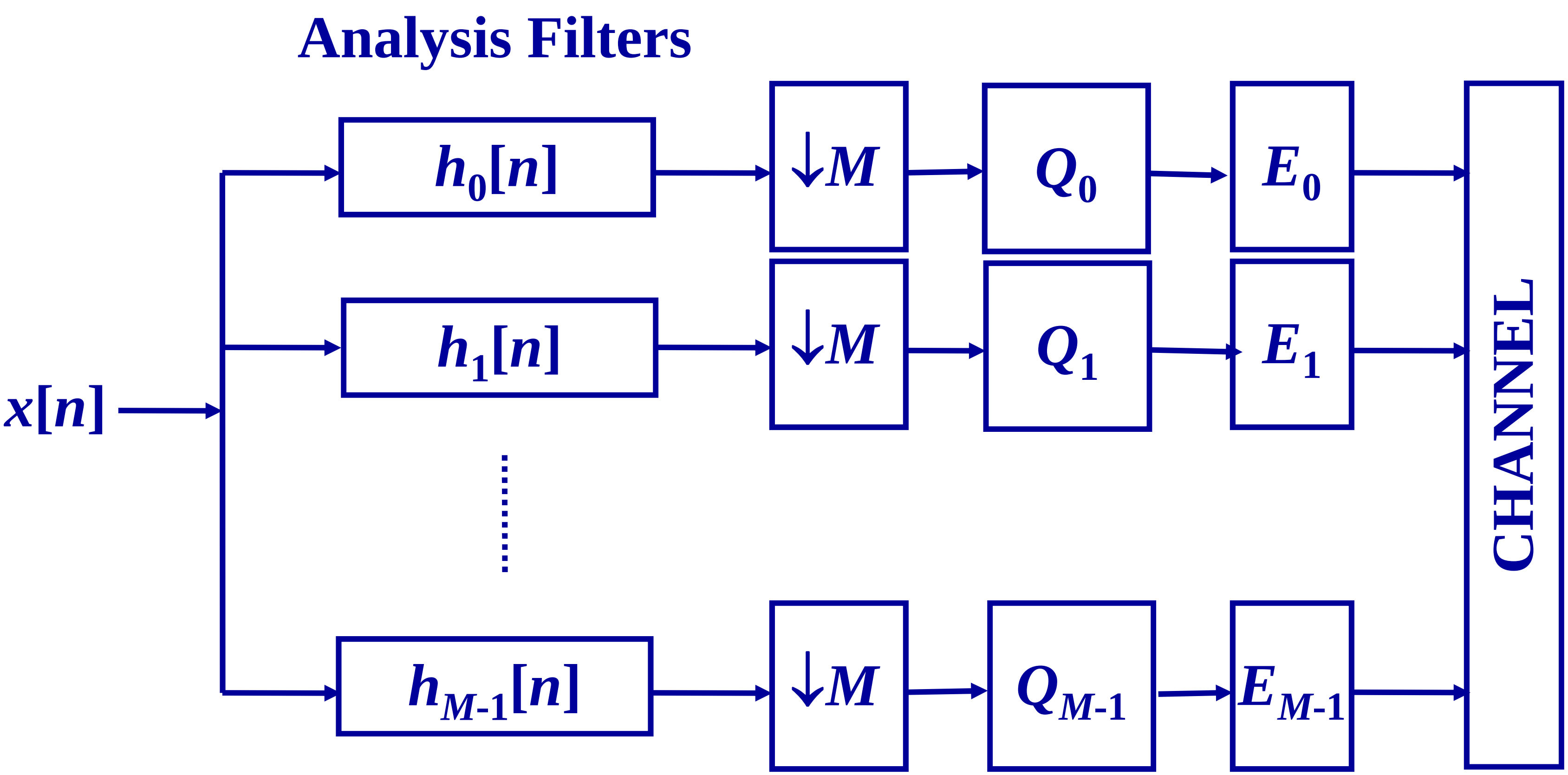
  The signal $x[n]$ can be recovered as:

  $$x[n-1]=-(y[n]-y[n-1])/2, \; x[n]=(y[n]+y[n-1])/2.$$

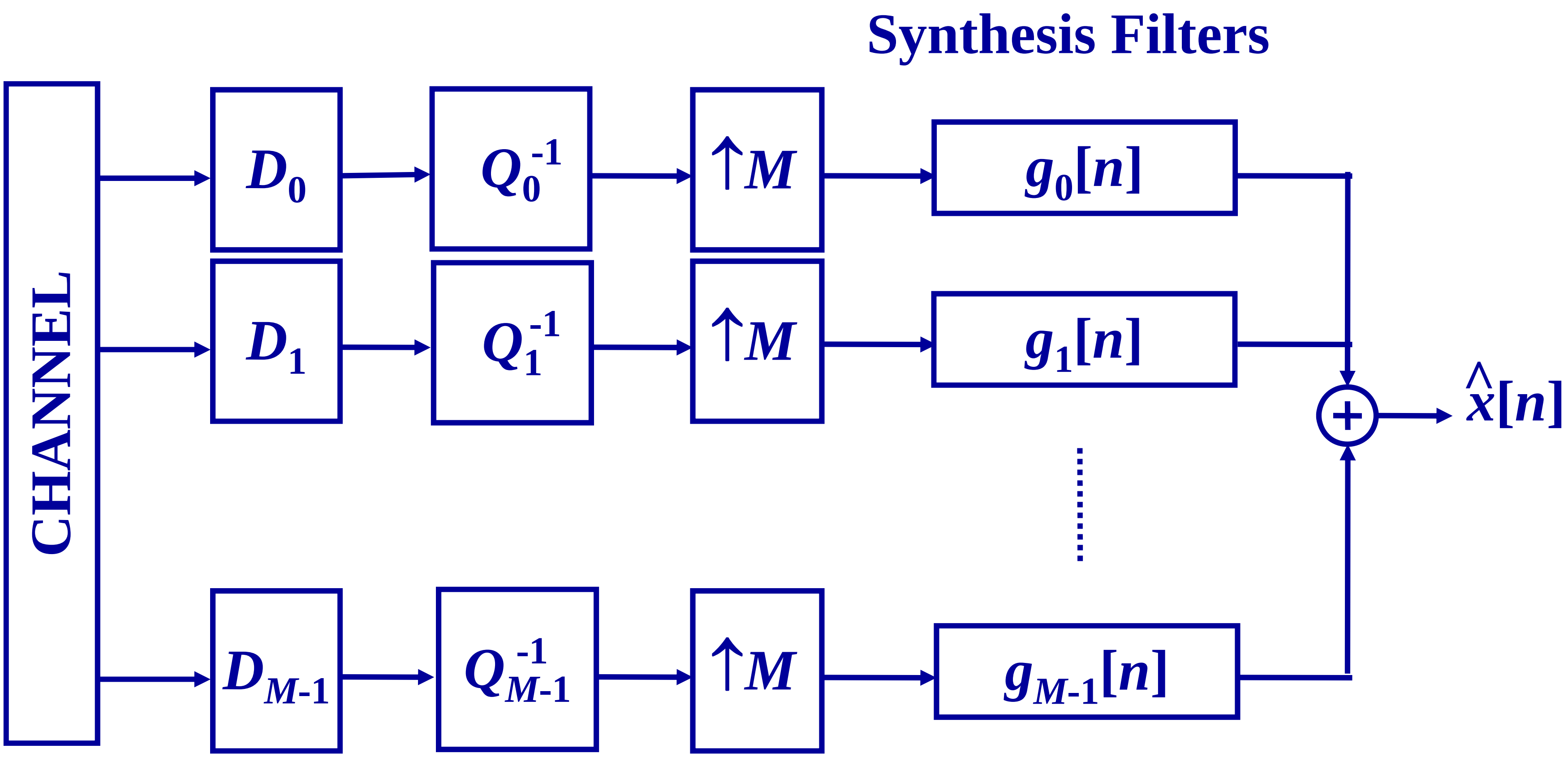
  The corresponding filters are:

  $$h[n]=\delta[n]+\delta[n-1], \quad g[n]=\delta[n]-\delta[n-1].$$

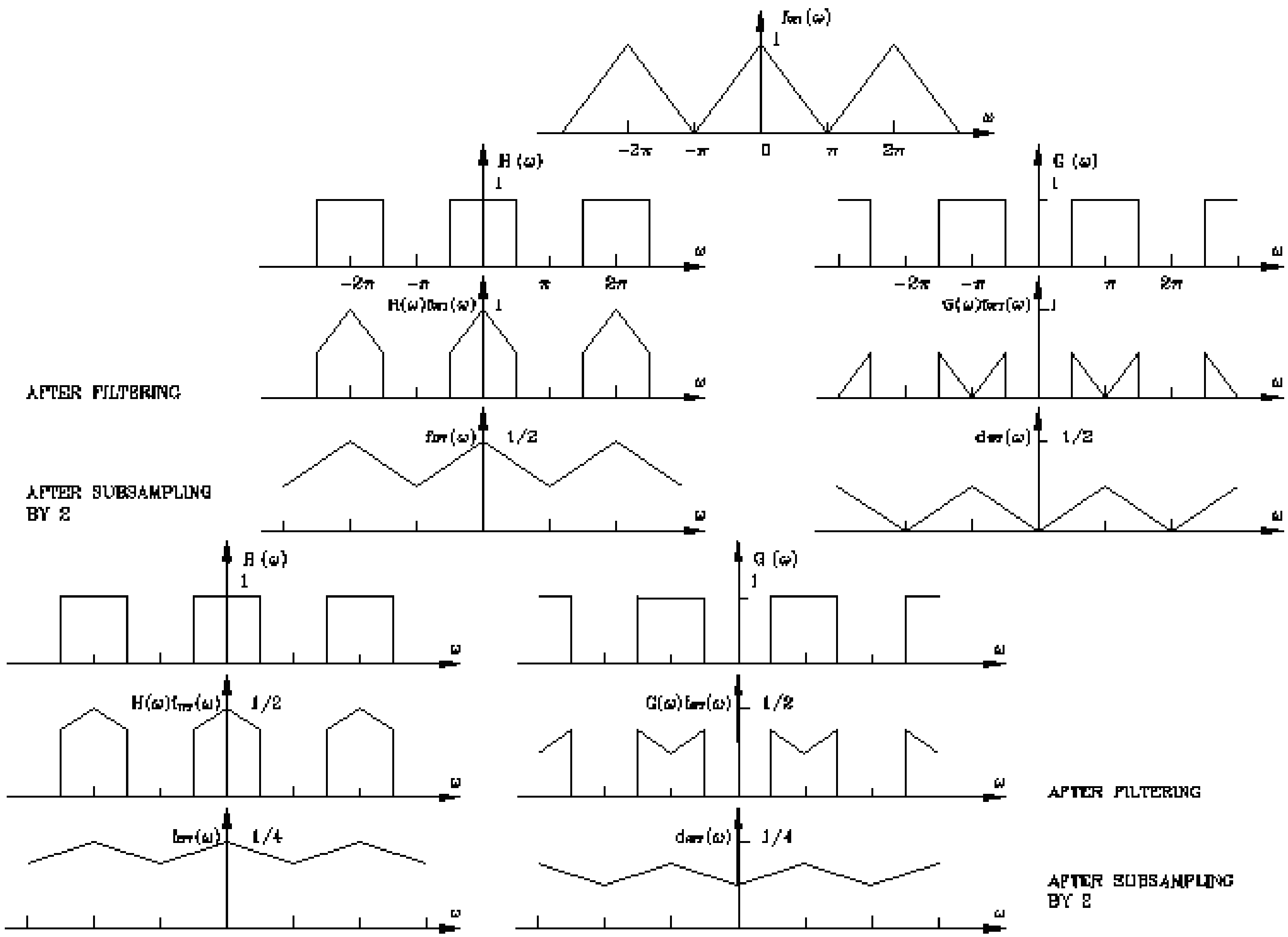
Subband Encoding



Subband Decoding



Filter Bank Decomposition



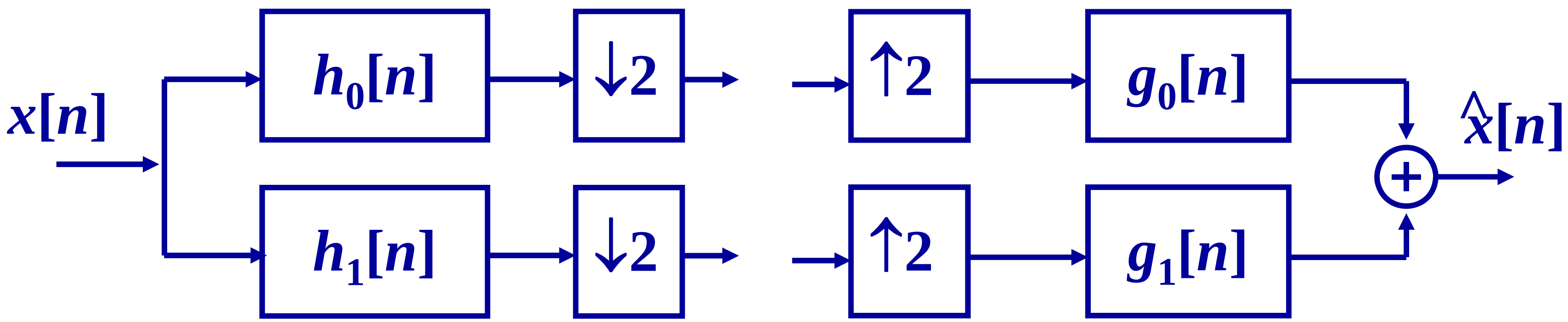
Subband Coding Design

- Design issues:
  - Filter banks design
  - Quantization
  - Entropy coding
  - Bit allocation among all subbands.
- The general form of digital filters is:

$$y[n] = \sum_{i=0}^{N} a_i x[n-i] + \sum_{i=1}^{M} b_i y[n-i]$$

- Filters can be analyzed in the time domain (convolution), frequency domain (Fourier transformation), or Z domain (transfer functions.)

Two-Band Filter Banks



- For M=2 the filters are easy to analyze.
- Goal:
  - good frequency-domain separation
  - no aliasing terms
  - perfect reconstruction: system is equivalent to a delay

Two-Band Filter Banks (*cont.*)

- Quadrature Mirror Filters (QMF, Croisier, Esteban, Galand 1976, Johnston 1980):

  symmetric: $h_0[n]=h_0[N\text{-}1\text{-}n]$,

  also satifiy: $h_1[n]=(-1)^n h_0[n]$,

  $g_0[n]=h_0[n]$,

  $g_1[n]=-(-1)^n h_0[n]$.

- Properties:
  - No aliasing, no magnitude distortion, some phase distortion.
  - The decomposition efficiency (i.e. frequency domain separation) increases with the length of filters.
  - Efficient polyphase implementation.

Two-Band Filter Banks (*cont.*)

- Conjugate Quadrature Filters (CQF, Smith-Barnwell, 1984):

$$h_1[n]=(-1)^n\,h_0[L-1-n],$$
$$g_0[n]=h_0[L-1-n],$$
$$g_1[n]=-(-1)^n h_0[n].$$

- Properties:
  - Perfect reconstruction, i.e. $x[n]=\hat{x}[n+m]$, where m is a linear delay.
  - Better frequency characteristics for the same number of taps comparing to the Johnston filters.
  - Closely related to wavelets.

Two-Band Filter Banks (*cont.*)

- 9/7 biorthogonal FIR filter bank (Cohen, Daubechies and Feauveau, 1992):

$$g_1[n]=(-1)^{n+1} h_0[-1+n],$$
$$h_1[-n]=(-1)^{n+1} g_0[1-n],$$
$$\sum_n h_0[-n] g_0[n+2k]=\delta[k].$$

- Properties:
  - Perfect reconstruction,
  - Symmetric, linear phase,
  - Most popular filter bank in image compression.

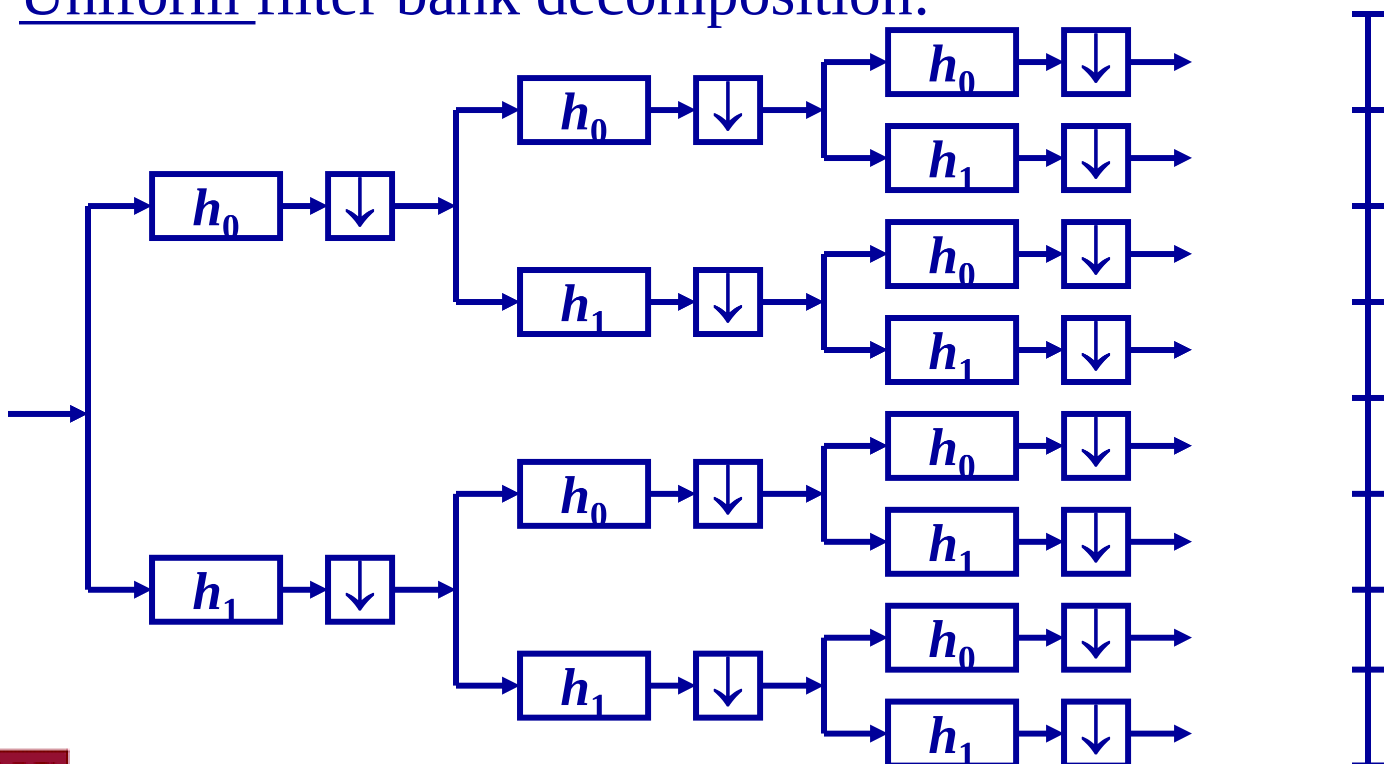
Two-Band Filter Banks (*cont.*)

- 9/7 biorthogonal FIR filter bank:

n	$h_0[n]$	$h_1[n]$	$g_0[n]$	$g_1[n]$
-4	**0.037829**	**0**	**0**	**0**
-3	**-0.023849**	**0**	**-0.064539**	**0.037829**
-2	**-0.110624**	**0.064539**	**-0.040690**	**0.023849**
-1	**0.377403**	**-0.040690**	**0.418092**	**-0.110624**
0	**0.852699**	**-0.418092**	**0.788485**	**-0.377403**
1	**0.377403**	**0.788485**	**0.418092**	**0.852699**
2	**-0.110624**	**-0.418092**	**-0.040690**	**-0.377403**
3	**-0.023849**	**-0.040690**	**-0.064539**	**-0.110624**
4	**0.037829**	**0.064539**	**0**	**0.023849**
5	**0**	**0**	**0**	**0.037829**

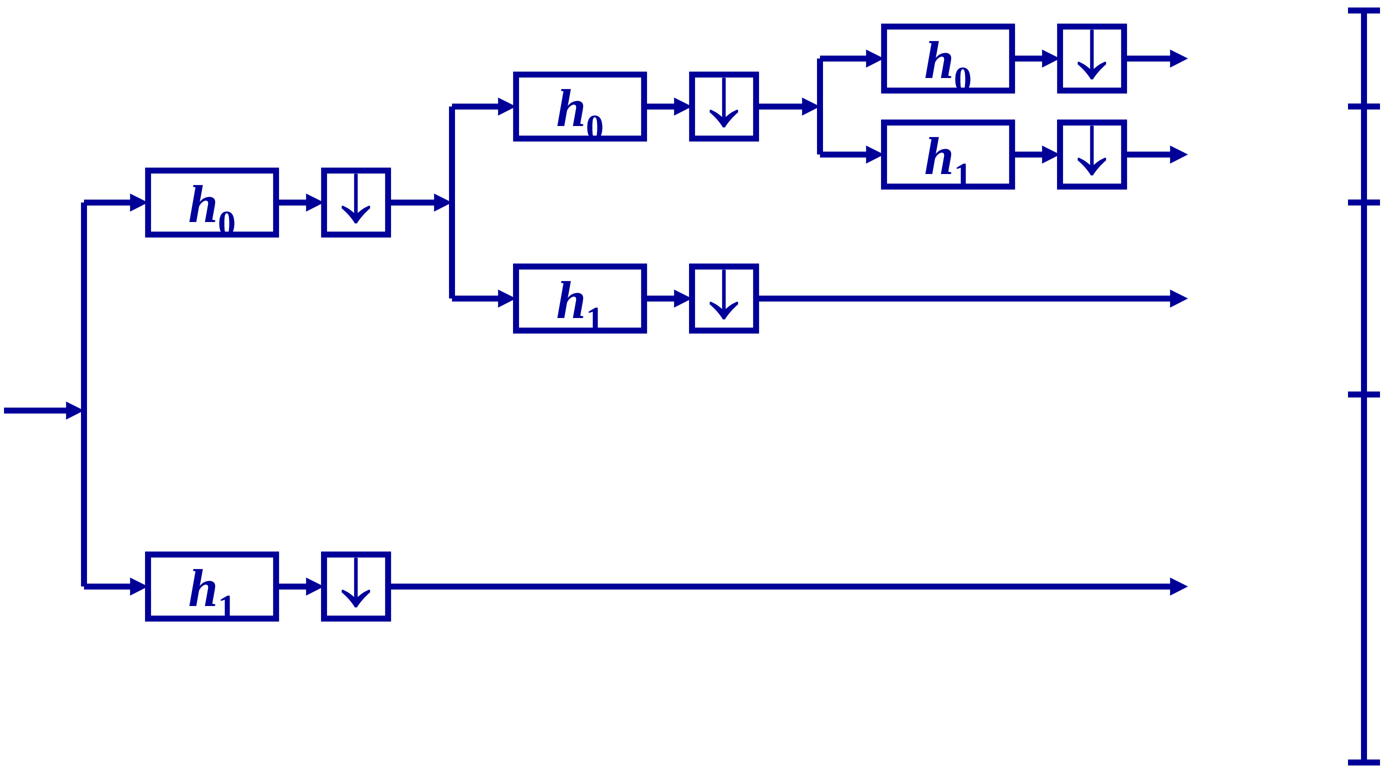
Tree-Structured Filter Banks

- We can design an M-band filter bank by successively applying the **2**-band filter banks.
- Uniform filter bank decomposition:



Tree-Structured Filter Banks (*cont.*)

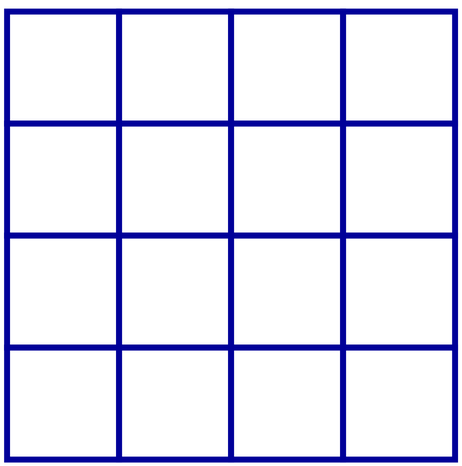
- Octave Band filter bank decomposition

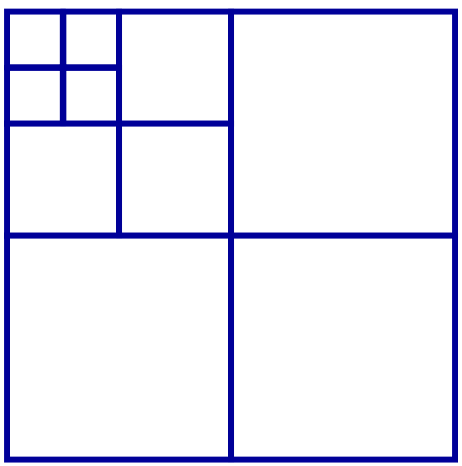


2-D Tree-Structured Filter Banks

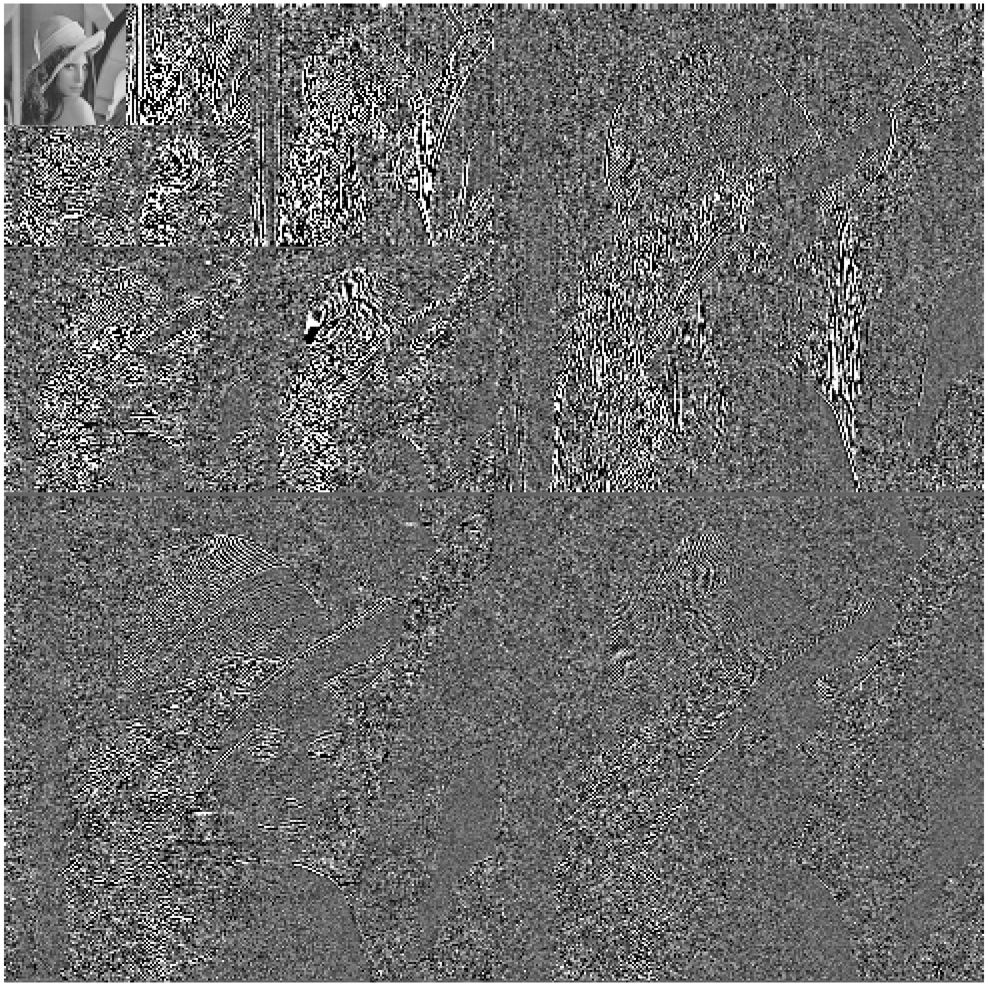
- Most 2-D filter banks are obtained by applying 1-D decompositions separably.

Uniform decomposition   Octave-tree decomposition





Example of Subband Decomposition



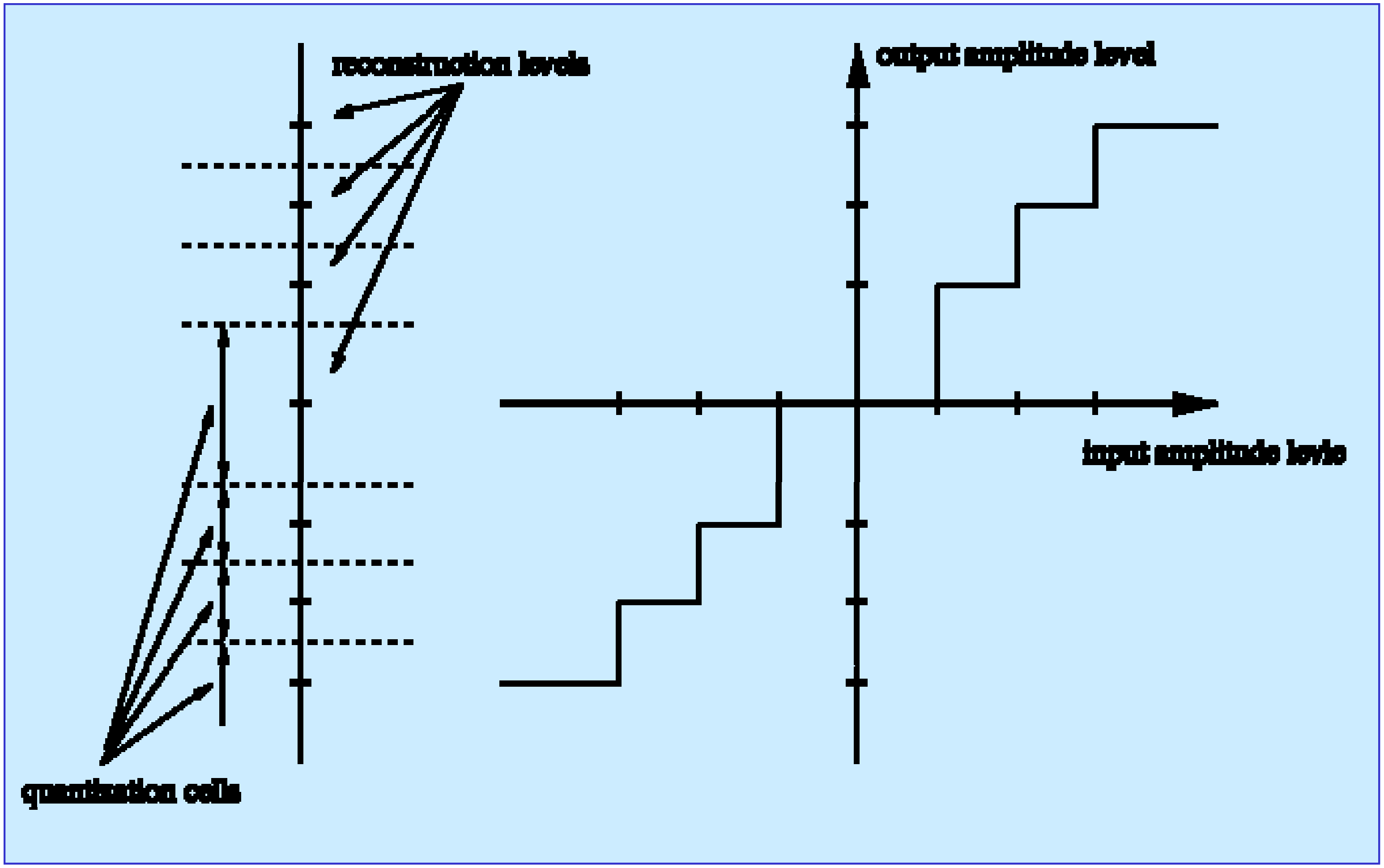
Quantization

- Quantization:

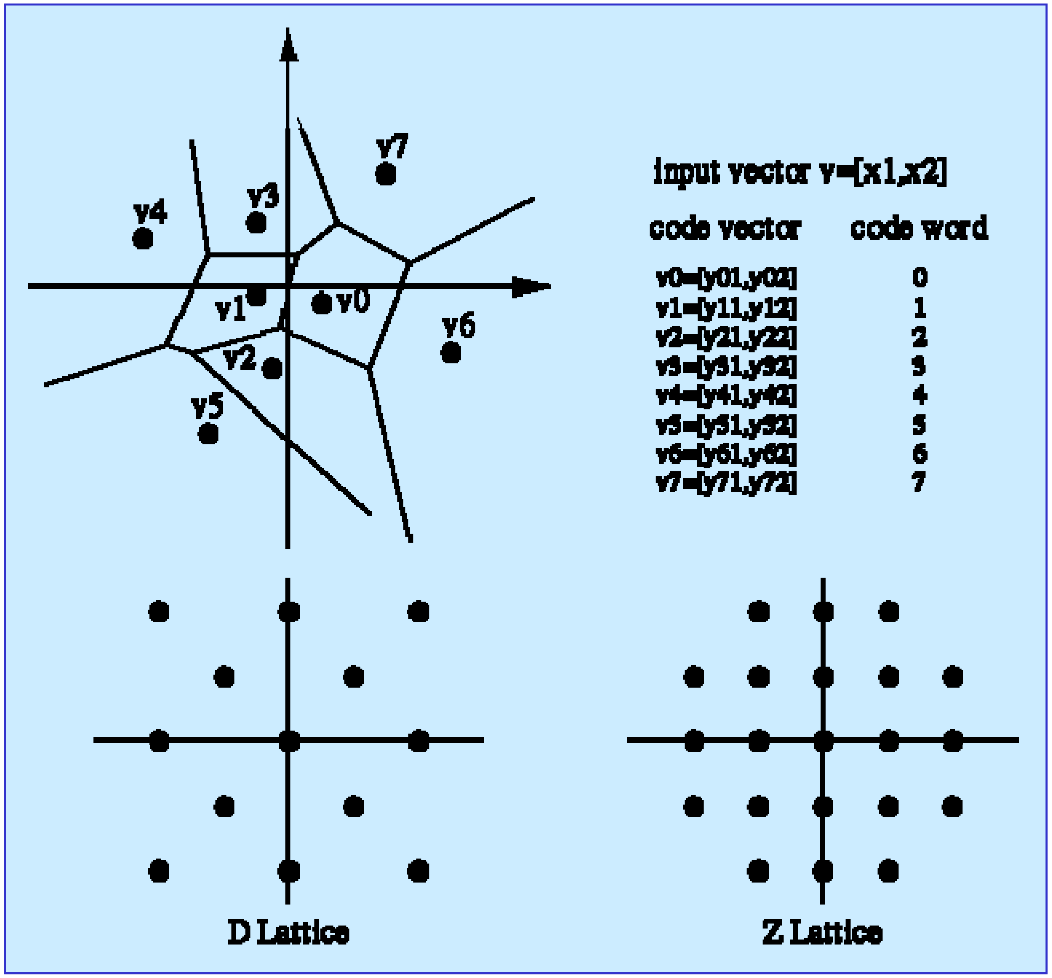
  map a large number of input amplitude levels to a small number of output amplitude levels with non-recoverable loss of quality.

- Why:
  - reduced level of amplitude $\Rightarrow$ reduced number of bits required to represent each pixel.
- How:
  - scalar quantization (SQ),
  - vector quantization (VQ).

Example of SQ



Examples of VQ



Scalar Quantization

- The quantized $\hat{x}[n]$ is a discrete source of symbols with probabilities

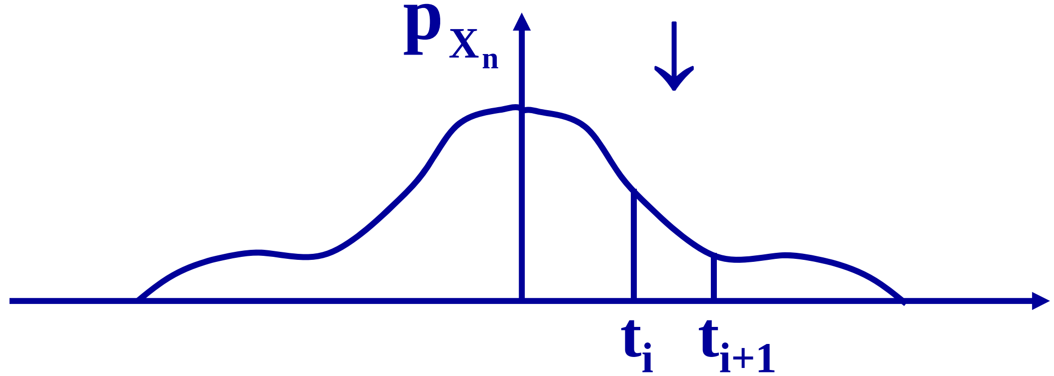
$$P[S_i] = \int_{t_i}^{t_{i+1}} p_{X_n}(x_n)\,dx_n$$

$$t_i \le x[n] \le t_{i+1}$$

$$\Downarrow$$

$$\hat{x}[n] = S_i$$

$$\downarrow$$



Scalar Quantization (*cont.*)

- An N-point quantizer is a mapping

$$Q : \Re \rightarrow C = \{y_1, ..., y_N\} \subset \Re$$

where C is the set of output points or reproduction values.

- A quantizer is completely specified by its codebook C and the partition of the real axis into cells

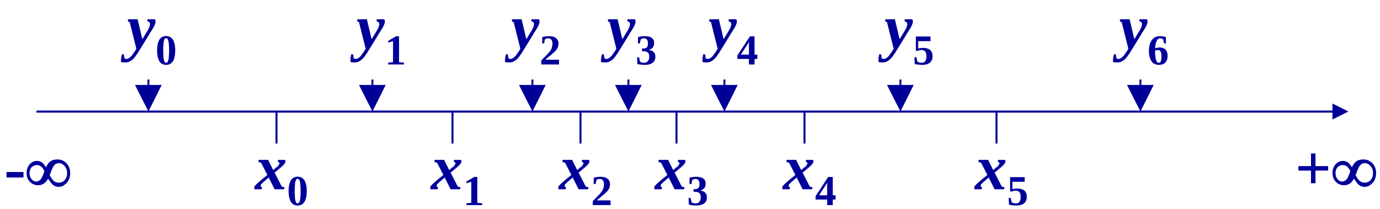
$$R_i = \{x \in \Re; Q(x) = y_i\} = Q^{-1}(y_i)$$

note that $\cup R_i = \Re$

$$R_i \cap R_j = \emptyset \text{ for } i \neq j$$

Regular Scalar Quantizers

- We are only concerned with regular quantizers for which
  - the cells $\mathbf{R_i}$ are intervals of the form (x_{i-1}, x_i) together with one or both of its endpoints.
  - The reconstruction levels $y_i \in \mathbf{R_i}$.

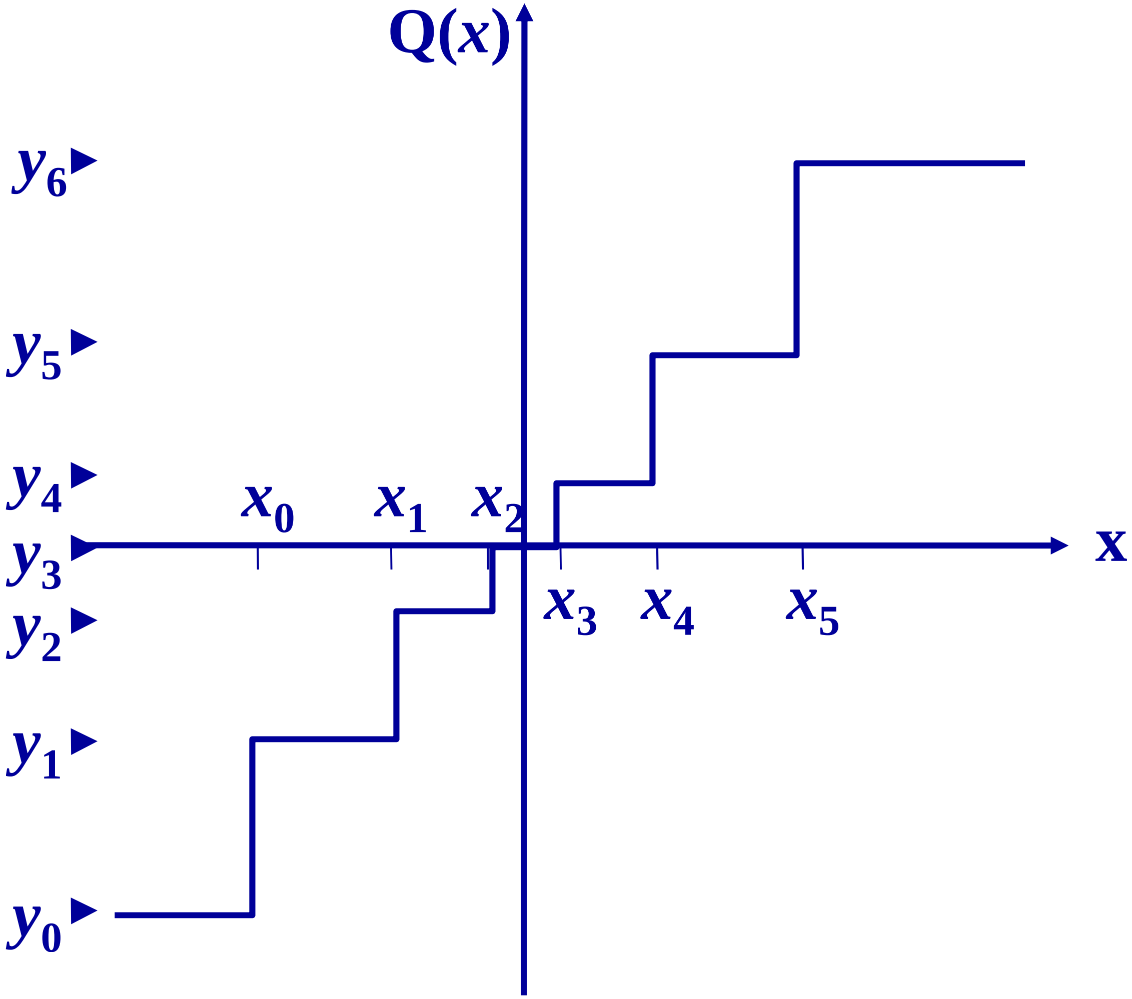


Regular Scalar Quantizers (*cont.*)

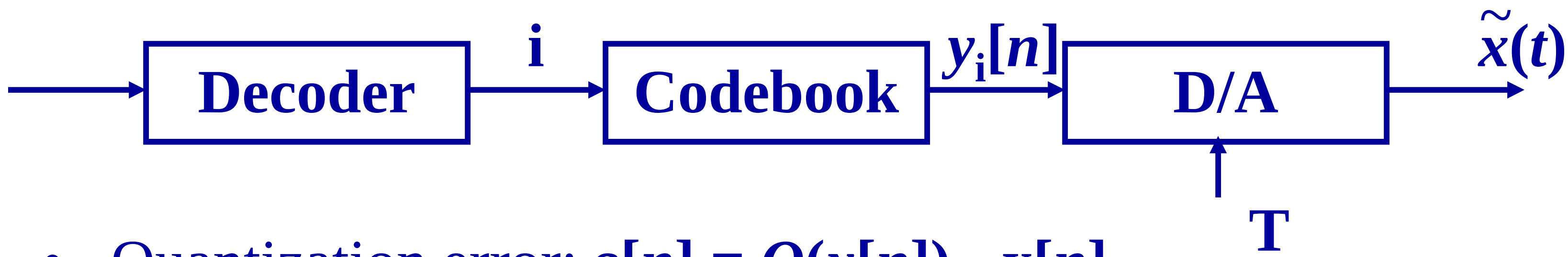
Overload region

Granular region

Overload region



Inverse Quantization



- Quantization error: **ε[*n*] = *Q*(*x*[*n*]) - x[*n*]**
- Any distortion measure can be used. The most widely used is the MSE:

$$D = E[(X - Q(X))^2] = \int_{-\infty}^{+\infty} (x - Q(x))^2 f_X(x)dx$$

- For regular quantizers:

$$D = \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} (x - y_i)^2 f_X(x)dx$$

Uniform Quantizers

- Regular quantizers with:
  - same size partitions in the granular region
  - reconstruction levels are the midpoints of the cells
- The most popular and simple
- According to the way they handle $\boldsymbol{x} = \mathbf{0}$:
  - midtread quantizer: zero is one of the reconstruction levels.
  - midrise quantizer: zero is one of the thresholds.
- Minimizes maximum distortion.
- For uniform sources, it is optimum for any distortion.

Uniform Quantizers (*cont.*)

- For uniform distributions, the quantization error $\boldsymbol{\varepsilon} = \mathbf{Q}(\boldsymbol{x}) - \boldsymbol{x}$ of an N-point uniform quantizer is uniformly distributed in the interval $[-\Delta/2, +\Delta/2]$, where $\Delta = 2x_{max}/N$.
- Then $E[\varepsilon] = 0$, $E[\varepsilon^2] = \Delta^2/12$, $E[|\varepsilon|] = \Delta/4$.
- If the quantizer output is encoded using **n** bits per sample:

$$SNR(dB) = 10\log_{10}\frac{\sigma_x^{\,2}}{\sigma_\varepsilon^{\,2}} = 10\log_{10} N^2 = 6.02\log_2 N = 6.02n$$

- If *pdf* is not uniform, an optimal quantization step size can be found that minimizes **D**.

Optimum Quantizers

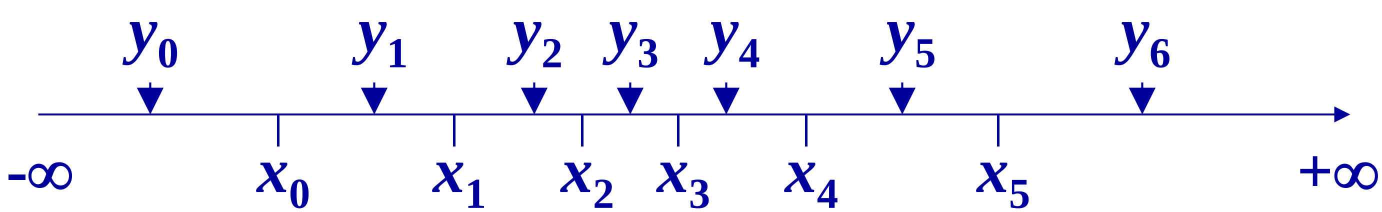
- Non-uniform quantizers attempt to decrease the average distortion by assigning more quantization levels to more probable regions.
- For given N and input pdf $f_X(x)$, we need to choose $\{x_1,\ldots, x_N\}$ and $\{y_1,\ldots, y_N\}$ that minimize the distortion (MSE)

$$D = \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} (x - y_i)^2 f_X(x)dx$$

- It is usually difficult to solve this optimization problem with $2N$ variables.

Optimum Quantizers (*cont .*)

- It is easier to find the optimal solutions under these conditions:
  - For a given interval partition (x_{i-1}, x_i) (encoder), design the optimum reconstruction levels y_i (decoder);
  - For a given reconstruction levels y_i (decoder), design the optimum interval partition (x_{i-1}, x_i) (encoder).
- When both the reconstruction levels and the partition are optimal, we have a (locally?) optimal quantizer.



The Optimum Codebook

- For a given interval partition (x_{i-1}, x_i), to minimize the MSE:

$$D = \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} (x - y_i)^2 f_X(x)dx$$

The y_i that minimizes

$$\int_{x_{i-1}}^{x_i} (x - y_i)^2 f_X(x)dx = E\left[(x - y_i)^2 \mid x \in R_i\right]$$

is the centroid of $R_i = (x_i, x_{i-1})$:

$$y_i^* = E\left[x \mid x \in R_i\right] = \frac{\int_{x_{i-1}}^{x_i} x f_X(x)dx}{\int_{x_{i-1}}^{x_i} f_X(x)dx}$$

The Optimum Partition

- If the codebook (*i.e.* the set of reconstruction levels) is fixed, for any distortion measure, the optimum partition cell R_i is defined by the nearest neighbor encoder rule:

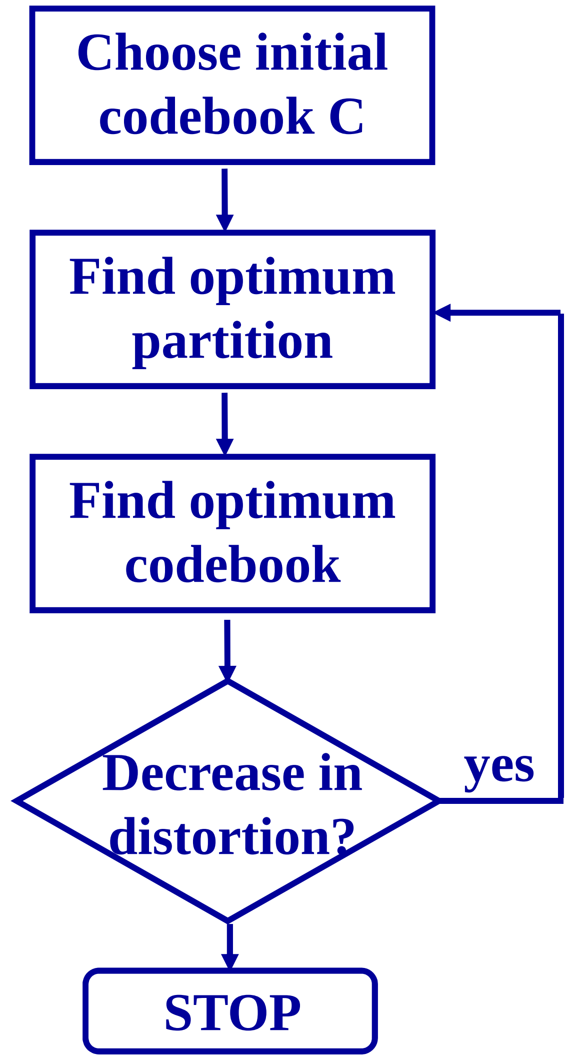
$$R_i = \{x: d(x, y_i) \le d(x, y_j) \text{ for all } j \ne i\}$$

- For regular quantizers and "normal" distortion function, the optimum thresholds (*i.e.* cell boundaries) are:

$$x_{i-1} = \frac{y_{i-1} + y_i}{2}$$

The Lloyd Algorithm

- It is difficult to solve both sets of equations analytically.
- An iterative algorithm known as the Lloyd algorithm solves the problem by iteratively optimizing the encoder and decoder until both conditions are met with sufficient accuracy.



Implementing the Lloyd Algorithm

- When the *pdf* is known, the centroids can be computed using numerical integration.
- In practice, the *pdf* is usually not known. A sufficiently large representative set (training set) of input samples is used, and statistical averages are replaced by set averages. The centroid of the set $\boldsymbol{R_i}$ is computed as

$$y_i = \frac{1}{N_i} \sum_{x_i \in R_i} x_i$$

where $\boldsymbol{N_i}$ is the number of training samples in $\boldsymbol{R_i}$.

Properties of Lloyd Quantizers

- $E[Q(X)] = E[X]$ (mean preservation)
  where X is quantizer input and $Q(X)$ is output.
- $E[Q(X)\bullet(Q(X)-X)] = 0$ (orthogonality)
  where $(Q(X)-X)$ is the quantization noise.
- $E[(Q(X)-X)^2] = \sigma_X^2 - \sigma_{Q(X)}^2$
  where σ_X^2 is the variance of the input,
  $\sigma_{Q(X)}^2$ is the variance of the output.
- $\sigma_{Q(X)}^2 \leq \sigma_X^2$
- $E[X\bullet(Q(X)-X)] = -\sigma_{Q(X)}^2$

Vector Quantization

- VQ is a generalization of scalar quantization. It is a mapping from a $\boldsymbol{k}$-dimensional real value vector to a codeword

$$Q : \Re^k \to C = \{\underline{y}_1, \ldots, \underline{y}_N\} \subset \Re^k$$

- $\boldsymbol{C}$ is called the codebook and has size $\boldsymbol{N}$. Each of its element is a $\boldsymbol{k}$-dimensional real value vector.
- Associated with an $\boldsymbol{N}$-point VQ is a partition of the k-dimensional real space into $\boldsymbol{N}$ regions (cells) given by:

$$R_i = \{\underline{x} \in \Re^k ; Q(\underline{x}) = \underline{y}_i\} = Q^{-1}(\underline{y}_i)$$

$$\text{where} \quad \cup R_i = \Re^k, \; R_i \cap R_j = \emptyset \;\text{ for }\; i \neq j$$

Vector Quantization (*cont.*)

- If the output of the VQ codebook is fixed-rate coded, then the rate of the VQ is $\boldsymbol{r = (\mathrm{Log}_2\, N)/k}$.
- Definition: A Vector Quantizer (VQ) is called <u>regular</u> if:
  - Each cell $\boldsymbol{R_i}$ is a convex set.
  - For each $\boldsymbol{i}$, the corresponding code vector belongs to $\boldsymbol{R_i}$.
- The encoder needs to know the geometry of the space partition. The encoding involves searching the nearest code vector in the set of all code vectors.
- The decoder needs to know the codebook (*i.e.* code words and code vectors). The decoding is usually a table-lookup.

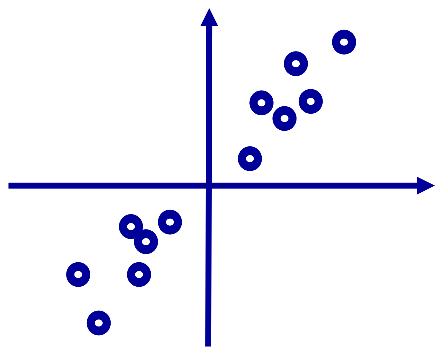
Vector Quantization (*cont.*)

- An example of a 2-D Vector Quantizer:
  - A map of the city that is divided into school districts.
  - Each school in its district represents the code vector.
  - Input: Location of child's residence.
  - Output: Rule by which each child is assigned to a school.

- Shannon's lossy source coding theorem states that: Given a signal vector (or data vector), no other coding technique exists that can outperform vector quantization.

Vector Quantization (*cont.*)

- The advantages of the VQ:
  - Exploit linear dependencies (correlation) and non-linear dependencies,



  - Better multi-dimensional space filling.